Theoretical Computer Science 410 (2009) 5403–5414

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Covering arrays avoiding forbidden edges

Peter Danziger [a], Eric Mendelsohn [b], Lucia Moura [c,*], Brett Stevens [d]

[a] *Ryerson University, Canada*
[b] *University of Toronto, Canada*
[c] *University of Ottawa, Canada*
[d] *Carleton University, Canada*

A B S T R A C T

Covering arrays (CAs) can be used to detect the existence of faulty pairwise interactions between parameters or components in a software system. The generalization considered here applies to the situation in which some input combinations are invalid, a requirement quite common in software testing. In this paper, we study covering arrays avoiding forbidden edges (*CAFE*s), where certain pairwise interactions are forbidden while all others must be covered, and we aim to minimize the number of tests. We establish a theoretical framework for this problem, by providing connections to the edge clique covering problem, lower and upper bounds, complexity results and a recursive construction. We also give an algorithm for the case of binary alphabets.

1. Introduction

An extended abstract of this paper has appeared [13]. This paper differs from that abstract by providing the proofs of the central results, stating some results in a more general form, more fully exploring the connection to the edge clique covering number on graphs and to error locating arrays, and discussing asymptotics.

This paper addresses the application problem of testing a complex system whose behavior depends on the values of k parameters or *factors*. Suppose each of the k factors may take any of g values. To exhaustively test the system, we would need g^k tests, which is too costly in practice, even for a moderate number of factors k. Instead, covering arrays have been largely used in this context since they offer effectiveness at a substantially lower cost, having applications ranging from software and hardware testing [5,12,19,22,23,27,36] to genomics [37] and material sciences [7]. In realistic situations, constraints involving the factors will restrict certain configurations of the parameters. In this paper, we investigate test plans given by covering arrays that avoid some fixed set of configurations of the parameters.

Covering arrays provide an alternative to exhaustive testing, by offering a much smaller test suite that guarantees coverage of all possible interactions from any t factors. In this paper, we focus on $t = 2$, where pairwise interactions of factors are tested. A *covering array* (CA) is a matrix with symbols from a g-ary alphabet g, with n rows and k columns. Each of its columns represents a parameter and each of its rows gives a test to be performed. The number of rows, n, is called the *size* of the array. The array must offer a pairwise ($t = 2$) coverage of factor values, that is, for any pair of the k factors, the corresponding columns exhaustively cover all possible g^2 combinations of values. In practice, small interaction coverage ($t = 2, 3, 4$) has been shown very powerful for software failure detection, and in particular $t = 2$ offers an excellent compromise between failure detection and size. Empirical results show that testing all pairwise interactions in a software

* Corresponding address: University of Ottawa, School of Information Technology and Engineering, 800 King Edward St, P.O. Box 450, Stn A, K1N 6N5 Ottawa, ON, Canada Tel.: +1 613 562 5800.
E-mail address: lucia@site.uottawa.ca (L. Moura).

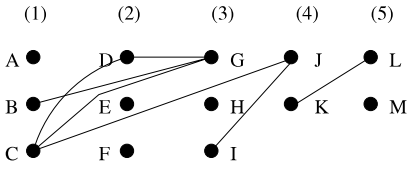
0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.07.057

Product line options (factors):	1) display	2) email viewer	3) camera	4) video camera	5) video ringtones
possible values:	A=16 Million colours B=8 Million colours C=Black and White	D=graphical E=text F=none	G=2 Megapixels H=1 Megapixel I=none	J=Yes K=No	L=Yes M=No

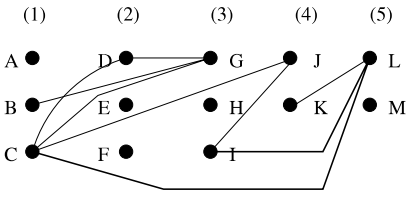
Constraints on valid configurations: Forbidden edges:

(C1) Graphical email viewer **requires** a colour display $\{C, D\}$
(C2) 2 Megapixel camera **requires** a color display $\{C, G\}$
(C3) Graphical email viewer **not supported** with the 2 Megapixel camera $\{D, G\}$
(C4) 8 Million colour display **does not support** a 2 Megapixel camera $\{B, G\}$
(C5) Video camera **requires** a camera and a colour display $\{I, J\}, \{C, J\}$
(C6) Video ringtones **cannot occur** with No video camera $\{L, K\}$

Graph G for given constraints:



Graph $\hat{G}$ for given and implied constraints:



Pairwise testing without constraints:

	display	email viewer	camera	video camera	video ringtones
test 1	A	F	I	J	L
test 2	B	E	I	K	M
test 3	A	E	G	K	M
test 4	C	F	G	K	L
test 5	B	F	H	J	M
test 6	A	D	I	J	M
test 7	C	E	H	J	M
test 8	B	D	H	K	L
test 9	C	D	G	J	L

a

Pairwise testing with constraints in $\hat{G}$:

	display	email viewer	camera	video camera	video ringtones
test 1	A	D	I	K	M
test 2	A	E	G	K	M
test 3	A	F	G	J	L
test 4	A	D	H	J	L
test 5	B	D	H	K	M
test 6	B	E	I	K	M
test 7	B	F	H	J	M
test 8	C	E	H	K	M
test 9	C	F	I	K	M

b

Fig. 1. Mobile phone product line.

system finds most of its faults [12,22,23]; some authors also link pairwise coverage to good "code coverage" [6,14]. The minimal size of a covering array with k factors with g possible values per factor is denoted by $CAN(k, g)$. For fixed g, this number is known to asymptotically grow as $\frac{g}{2} \log k$ [16], offering a practical solution over exhaustively going through g^k tests.

There is a vast literature on covering arrays and their efficient constructions (see Colbourn's surveys [9,10] and Hartman and Raskin [19]), including generalizations of the definition presented above. There are situations in which considering t-way interactions for $t > 2$ gives additional testing benefits [22] over $t = 2$, so we can look at covering arrays of strength t [9]. When each factor i has g_i possible values, *mixed covering arrays* (MCAs) [32] have been considered. Another generalization of CAs and MCAs targets situations where some factor combinations need not be tested [31,30], the so called *covering arrays on graphs*.

Fig. 1 shows an example of a system with $k = 5$ factors. Two of the factors can have three possible values, while the other three can take any of two different values. Let us first consider the case without constraints on valid configurations. Testing exhaustively such a system would require $72 = 3 \times 3 \times 2 \times 2 \times 2$ tests, but an MCA of size 9 allows us to test all pairwise interactions (see MCA in Fig. 1a). If the system is susceptible to failure due to a single factor value or due to a combination of two factor values, at least one of the tests given by this MCA is guaranteed to fail.

In software, hardware and network testing, we often encounter extra constraints on feasible parameter combinations. Cohen et al. [8] discuss the application in highly configurable systems, where these types of constraints are common. The example in Fig. 1 with constraints (C1) to (C6) is extracted from [8]; this hypothetical example reflects portions of the Nokia 6000 series of mobile phones. These types of constraints can be translated into forbidden tuples (interactions) that cannot appear as part of a valid test. Other applications frequently present extra constraints such as drug interaction testing, where certain drug combinations may be forbidden due to known side effects of their combined use, and testing substance combinations in material science, where some material combinations are explosive and must not be tested [7].

The problem of constructing a test set that covers all allowed pairwise interactions while avoiding a given set of forbidden configurations or interactions has not been satisfactorily addressed in the literature nor by the combinatorial interaction

testing tools available [8]. Hartman and Raskin state it as a major open problem in covering arrays and suggest "There are many open theoretical and algorithmic problems which remain to be tackled. We hope that this paper will inspire others to improve on our results" [19]. The paper by Cohen et al. gives a comparison of various heuristic algorithms that incorporate such constraints [8]. Bryce and Colbourn and Ronneseth and Colbourn have considered these constraints in the context of heuristic constructions of CA [5,35]. They consider both edges that are strictly forbidden as well as edges which we strongly do not want but could be accepted if necessary. Finally Hou et al. state an equivalent problem when they consider aligning DNA sequences [20]. These references do not discuss the theory of avoiding configurations in any detail. The present paper concentrates on establishing initial theoretical results on the problem, algorithms for building covering arrays with forbidden configurations as well as bounds on their sizes.

In general, the forbidden interactions may be of any size; for instance, in the example of Fig. 1, we could add the following constraint: "(C7) The combination of 16 Million colours, Text and 2 Megapixel camera will **not be supported**" [8], which is equivalent to forbidding the interaction $\{A, E, G\}$. However, in the present paper we concentrate on the case of forbidden **pairwise** interactions like in constraints (C1) to (C6). In this case, the forbidden interactions can be represented by the edges of a graph as illustrated in Fig. 1. An array whose rows cover all the non-edges of the graph and avoid the edges of the graph is called a *Covering Array avoiding Forbidden Edges* (*CAFE*), and is exemplified for this case in Fig. 1b.

One might think of the following "simple" construction to build a $CAFE(n, G)$: Take a covering array for the case with no forbidden edges and for each row that contains a forbidden combination, replace it with one or more rows that avoid the forbidden edges and still cover the same desired pairs. It turns out that depending on the graph G, a $CAFE(n, G)$ may or may not exist and that in certain cases even determining which of these cases hold is NP-complete. Thus the simple act of "replacing it with one or more rows" may not be possible or could be possible but computationally difficult to do! This difficulty confounds the heuristics of [5,35] even though they may work in many cases. The existence and construction of *CAFE*s is a hard problem in general and simplistic methods may not always work.

Next, we summarize our contributions and the organization of the paper. In Section 2, we give the main definitions, basic bounds and establish relationships between *CAFE*s and error-locating arrays, and the problem of covering the edges of a graph by cliques. In Section 3, we study the complexity of the problem. In particular, we show that determining whether G admits a *CAFE* can be decided in polynomial time for $g = 2$, while it is NP-complete for $g \geq 5$. In Section 4, we give a general recursive construction for *CAFE*s based on *CAFE*s for the connected components of the graph. We also provide an asymptotic result based on number of factors and number of edges in the graph. In Section 5, we give an algorithm for $g = 2$ that partitions the edges of the original graph and gives a *CAFE* whose size is bounded by $k + 1$ plus the size of an edge clique covering number of a certain subgraph of the complement. In particular, when G is bipartite, our algorithm produces a covering array of a size which is at most $k + 2$.

2. Definitions and preliminaries

Let us define the testing problem more formally. Consider a system with k *factors* (parameters or components) $1, \ldots, k$. Factor i can take one of g_i possible *values*, which we consider w.l.o.g. to be in the set $\{0, \ldots, g_i - 1\}$, denoted by $[0, g_i - 1]$. A *test* is an assignment of values to factors, i.e., a k-tuple in $[0, g_1 - 1] \times \cdots \times [0, g_k - 1]$. The execution of a test can have two outcomes: *pass* or *fail*; we call it a *passing* or a *failing* test, respectively. An *interaction* is a set of values assigned to distinct factors: $I = \{(f_1, a_1), \ldots, (f_t, a_t)\}$, $f_i \neq f_j$ for $i \neq j$, and $a_i \in [0, g_{f_i} - 1]$, $1 \leq i \leq t$. An interaction I is a *t-way interaction* if $|I| = t$. We say that a test (or a k-tuple) $T = (T_1, \ldots, T_k)$ *covers* interaction $I = \{(f_1, a_1), \ldots, (f_t, a_t)\}$, if $T_{f_i} = a_i$ for $1 \leq i \leq t$. Thus, a test covers exactly $\binom{k}{t}$ t-way interactions, $1 \leq t \leq k$. We assume that failures are caused by faulty interactions, that is, the execution of a test fails if and only if it covers one or more faulty interactions. Covering arrays are combinatorial designs that correspond to test suites that cover all t-way interactions of factor values, and consequently all s-way interactions with $1 \leq s \leq t$.

Definition 1. A *mixed covering array*, A, denoted by $MCA(n; t, (g_1, \ldots, g_k))$, is an $n \times k$ array, such that each column i (corresponding to a factor) has values from the alphabet $[0, g_i - 1]$, and every possible t-way interaction is covered by some row, or in other words, for every t-set of factors $\{f_1, \ldots, f_t\}$ and every t-tuple of values $(a_1, \ldots, a_t) \in [0, g_{f_1} - 1] \times \cdots \times [0, g_{f_t} - 1]$, there exists at least one row r (corresponding to a test) such that $A[r, f_j] = a_j$ for all $j \in [1, t]$. Given t and $g_1, \ldots, g_k$, the *mixed covering array number*, denoted by $MCAN(t, (g_1, \ldots, g_k))$, is the smallest n for which an $MCA(n; t, (g_1, \ldots, g_k))$ exists. When $g_i = g$ for all $1 \leq i \leq k$, we call the object simply a *covering array* and simplify the notation to $CA(n; t, k, g)$, and $CAN(t, k, g)$.

The test suite in Fig. 1a is an example of $MCA(9; 2, (3, 3, 2, 2, 2))$. Since $g_1 g_2 = 9$ is a lower bound for n, this gives $MCAN(2, (3, 3, 2, 2, 2)) = 9$.

Consider a graph whose edges represent the forbidden pairwise interactions in a system like in Fig. 1. It turns out to be convenient for the theory that will follow to allow this graph to have loops. Let $G = G_{(g_1, \ldots, g_k)}$ denote a graph with k parts of sizes $g_1, \ldots, g_k$ that is k-partite except for the possible existence of loops. The vertices of G are v_{i,a_i}, indexed by i, a_i where $i \in [1, k]$ and $a_i \in [0, g_i - 1]$. If $g_1 = \cdots = g_k = g$, then we simplify the notation to $G = G_{k,g}$. We define a graph $G^{|}$ on the same vertex set as G and including the edges from $E(G)$ but also containing all the edges $\{v_{i,a}, v_{i,b}\}$ for $a \neq b \in [0, g_i - 1]$. A graph G is said to be *factor connected* if $G^{|}$ is connected; *factor-connected components* of G correspond to components of $G^{|}$.

To be able to discuss when a $CAFE(n, G)$ exists or even if it is possible to have a single test that avoids G, we need precise definitions of which pairs that are not in G can be covered whilst avoiding G itself. A k-tuple $T = (T_1, \ldots, T_k) \in [0, g_1 - 1] \times \cdots \times [0, g_k - 1]$ is said to *avoid* $G = G_{(g_1,\ldots,g_k)}$ if for all $i, j \in [1, k]$, we have $\{v_{i,T_i}, v_{j,T_j}\} \notin E(G)$. We say that an interaction $\{(i, a), (j, b)\}$, with $i \neq j$ if $a \neq b$, such that $\{v_{i,a}, v_{j,b}\} \notin E(G)$ is *consistent* with G if there exists a k-tuple T with $T_i = a$ and $T_j = b$ that avoids G. A graph is *consistent* if all interactions $\{(i, a), (j, b)\}$, with $i \neq j$ if $a \neq b$, where $\{v_{i,a}, v_{j,b}\} \notin E(G)$ are consistent. This definition is equivalent to saying that all forbidden interactions implied by the edges of the graph are also edges of the graph.

Definition 2 (*CAFE*). A *covering array with forbidden edges* for a graph $G = G_{g_1,\ldots,g_k}$ is an $n \times k$ array A with each column i having symbols from the alphabet $[0, g_i - 1]$, and denoted by $CAFE(n, G)$, such that

1. each row of A forms a k-tuple avoiding G;
2. for all $v_{i,a}, v_{j,b} \in V(G)$ with $i \neq j$, if $\{v_{i,a}, v_{j,b}\} \notin E(G)$, then there exists a row r such that $A_{r,i} = a$ and $A_{r,j} = b$.

We denote by $CAFEN(G)$ the minimum n for which there exists a $CAFE(n, G)$, if such an object exists, or $+\infty$ otherwise. Similarly, $CAFE^1(n, G)$ and $CAFEN^1(G)$ are defined by substituting the second condition above by a pointwise coverage requirement: for all $v_{i,a} \in V(G)$ such that $\{v_{i,a}, v_{i,a}\} \notin E(G)$, there exists a row r such that $A_{r,i} = a$.

We note that the definition of $CAFE(n, G)$ does not have to specifically mention loops because of the requirement that each row avoids G.

It is easy to see that there exists a *CAFE* for G if and only if G is consistent. The minimal supergraph of G that is consistent is called the *avoidance closure of* G and denoted by $\hat{G}$. So, a graph is consistent if and only if $G = \hat{G}$. Fig. 1 gives an example of G and $\hat{G}$. Let $E^{i,j}(G)$ denote the set of edges with an end in factor i and the other in factor j. When a *CAFE* exists it is also easy to calculate a lower and upper bound on its size,

$$\max_{1\leq i<j\leq k} (g_i g_j - |E^{i,j}(G)|) \leq CAFEN(G) \leq \sum_{1\leq i<j\leq k} (g_i g_j - |E^{i,j}(G)|), \tag{1}$$

since each nonedge between two factors must be covered and any row whose pairs are all covered elsewhere can be discarded.

As we add edges to $G = G_{k,g}$, $CAFEN(G)$ may decrease or increase. For fixed g, when G is empty we know $CAFEN(G) = CAN(k, g) \to \frac{g}{2} \log k$, as $k \to \infty$. The upper bound in Eq. (1) gives $CAFEN(G) \leq g^2\binom{k}{2} - |E(G)|$, which in the worst case is quadratic on k. Indeed, an $O(k^2)$ bound is best possible, as a graph consisting of two cliques induced by $\{v_{0,i} \in V(G) : 1 \leq i \leq \lfloor k/2 \rfloor\}$ and $\{v_{0,i} \in V(G) : \lfloor k/2 \rfloor + 1 \leq i \leq k\}$, respectively, requires $k^2/4$ rows just to cover the non-edges involving one vertex in each clique. In contrast, an $O(|E(G)|^2 + \log k)$ asymptotic upper bound is given in Corollary 11.

*CAFE*s are strongly related to Error-Locating Arrays (*ELA*s), which are arrays that can identify *and* locate all errors/failing interactions.

Definition 3 (*[28,29] (Error Locating Arrays for Graphs)*). Let $G = G_{(g_1,\ldots,g_k)}$. A row r of an array A is said to cover a pair of vertices $v_{i,a}, v_{j,b} \in V(G)$, if $A_{r,i} = a$ and $A_{r,j} = b$. A row of A is said to *locate* a pair of vertices of G if it covers that pair and all other pairs of vertices covered in the row are not edges of the graph. An *error-locating array* for G is an $n \times k$ array, A, with each column i having symbols from the alphabet $[0, g_i - 1]$, and denoted by $ELA(n, G)$, such that each pair of vertices $v_{i,a}, v_{j,b} \in V(G)$ with $i \neq j$ (edge or non edge) is located by some row of A.

In plainer terms, for a graph $G = G_{(g_1,\ldots,g_k)}$ and every pair of vertices not in the same factor, an $ELA(n, G)$ must have a row which covers the given pair and otherwise avoids the graph G. If there exists an $ELA(n, G)$ for some positive n then we say that G is *locatable*, and call the smallest such n the error-locating array number, $ELAN(G)$.

Theorem 3. *If $G = G_{(g_1,\ldots,g_k)}$ is locatable, then $CAFEN(G) = ELAN(G) - |E(G)|$.*

Proof. A minimal $ELA(n, G)$ can be decomposed into two sets of rows. One set whose rows cover each edge from G exactly once in one (distinct) row. Another set which avoids G and covers all non-edges. The first set must have exactly as many rows as there are edges in G and the second set is easily seen to be precisely a minimal $CAFE(n', G)$. □

*CAFE*s are also closely related to edge clique coverings of graphs.

Definition 4. Let G be a graph. An *edge clique cover* of G is a set of cliques $\{K_i\}_{i\in I}$ of G such that for every $e \in E(G)$ there exists an $i \in I$ such that $e \in K_i$. The *edge clique covering number* of G, $\theta'(G)$ is defined as the size of a minimum edge clique cover. We define an *edge clique covering* of G to be k-uniform if all the cliques have size k, and denote by $\theta'_k(G)$, the size of a minimum k-uniform edge clique cover of G, or $+\infty$ if one does not exist.

This relationship was first noticed by Hou et al. [20] although they only imply the restriction to k-uniform edge clique covers while never actually formally requiring it and thus confuse and obscure the differences between the two problems of *CAFE* and edge clique covers. In their paper they do not attempt to attack the *CAFE* problem except when the graph is empty and thus only produce results on normal covering arrays. Ronneseth and Colbourn [35] also briefly discuss the connection to the edge clique cover problem. Note that $\theta'_k(G)$ is different from k-clique edge covers that have been considered in the literature, where the cliques are not necessarily subgraphs of G [17].

It is easy to see that a $CAFE(n, G)$ with k factors corresponds to a k-uniform edge clique cover of $\overline{G^|}$, the complement of $G^|$, with size n.

Proposition 5. *If $G = G_{(g_1,\ldots,g_k)}$, then $CAFEN(G) = \theta'_k(\overline{G}) \geq \theta'(\overline{G})$.*

While there are many results on $\theta'(G)$, most of them are not directly applicable to our problem of determining $\theta'_k(G)$. However, in Section 5 we show that $\theta'(\overline{G}) = \theta'_k(\overline{G})$, when $G = G_{k,2}$ (Theorem 15). Therefore, we list next some of the results on $\theta'(G)$, which can then be applied to build *CAFE*s with binary alphabets.

Clearly the set of all edges of G is itself an edge clique cover so $\theta'(G) \leq |E(G)|$. Erdös, Goodman and Pósa proved that that this value on the Turán graph is an upper bound on $\theta'(G)$.

Theorem 6 (*Erdös, Goodman and Pósa [15]*). *Let G be a graph on v vertices. There exists an edge clique cover of G of size no greater than $\lfloor v^2/4 \rfloor$. Furthermore such an edge clique cover exists that consists solely of edges and triangles.*

This bound can be improved by a result of Lovász.

Theorem 7 (*Lovász [24]*). *Suppose that $= |E(G)| = \binom{v}{2} - d$ and t is the greatest natural number so that $t^2 - t \leq d$. Then $\theta'(G) \leq d + t$.*

Brigham and Dutton show that

$$\theta'(G) \leq \theta(G)(v - \theta(G)), \qquad \theta'(G) \leq \beta'(G)(v - \beta'(G)), \qquad \theta'(G) \leq \alpha'(G)\beta'(G)$$

where $\theta(G)$ is the vertex clique cover number, $\beta'(G)$ is the edge independence number and $\alpha'(G)$ is the edge covering number [1]. The last inequality requires that G have no isolated vertices. Indeed Brigham and Dutton collect many relations of $\theta'(G)$ to other graph parameters in [1–4]. Gyárfás has shown that if a graph G on v vertices has no isolated vertices and no pair of equivalent vertices (connected vertices with precisely the same neighbours other than themselves), then its edge clique-cover number is at least $\lceil \log_2(v+1) \rceil$ [18]. Since $\theta'(G) \leq \theta'_k(G)$, if $G = \hat{G} = G_{(g_1,\ldots,g_k)}$ and the complement has no equivalent vertices then $CAFEN(G) \geq \log_2(v+1)$ where $v = \sum_{i=1}^{k} g_i$.

It is known that determining $\theta'(G)$ is NP-Hard [21,34]. Moreover, the following hardness of approximation result has been established.

Theorem 8 (*Lund and Yannakakis [25]*). *There exists a $\delta > 0$ such that there does not exist a polytime approximation algorithm for $\theta'(G)$ that achieves a ratio of v^δ unless $P = NP$.*

This suggests that even approximating $CAFEN(G)$ might be a hard problem.

3. Computational complexity results

In this section, we establish main computational complexity results for the problems under study (Theorem 9). Consider the following languages, associated to decision problems of interest to us:

- AVOID$= \{< G = G_{(g_1,\ldots,g_k)} >:$ *there exists a k-tuple avoiding* $G\}$;
- ONE-COVER&AVOID$= \{< G = G_{(g_1,\ldots g_k)} >:$ *for some n there exists a CAFE*$^1(n, G)\}$;
- COVER&AVOID$= \{< G = G_{(g_1,\ldots,g_k)} >:$ *for some n there exists a CAFE*$(n, G)\}$;
- CAFEN$= \{\langle G = G_{(g_1,\ldots,g_k)}, N\rangle :$ there exists a *CAFE*$(N, G)\}$.

For each language L defined above, we define g-$L = \{< G = G_{(g_1,\ldots,g_k)} > \in L : g_1 = \cdots = g_k = g\}$.

Theorem 9. *Consider the decision problems defined above. Then,*

1. 2-AVOID *is in P.*
2. g-AVOID *is NP-complete for $g \geq 3$, and so* AVOID *is NP-complete.*
3. 2-ONE-COVER&AVOID *and* 2-COVER&AVOID *are in P.*
4. $(g-1)$-ONE-COVER&AVOID *and* g-COVER&AVOID *are NP-complete for $g \geq 5$.*
5. g-CAFEN *is NP-complete, for $g \geq 5$, and so* CAFEN *is NP-complete. (Note: this can now be obtained from the more recent theorem establishing that 2-CAFEN is NP-complete [26]).*

Proof. 1. Solve 2-AVOID by solving the following instance of 2-SAT. Associate to each vertex $v_{i,a}$ of G a literal $l_{i,a} = x_i$, if $a = 0$; $\neg x_i$, if $a = 1$. Associate to each edge $\{v_{i,a}, v_{j,b}\}$ of G a clause: $(\neg l_{i,a} \vee \neg l_{j,b})$. The conjunction of these clauses is an instance of 2-SAT$\in P$.

2. To prove that 3-AVOID is NP-complete, we reduce from 3-SAT. Let $\phi = (l_{1,0} \vee l_{1,1} \vee l_{1,2}) \wedge \cdots \wedge (l_{k,0} \vee l_{k,1} \vee l_{k,2})$ be a formula with k clauses with 3 literals each. Build G, a k-partite graph with 3 vertices per part, corresponding to the literals in each of the clauses, and such that $\{v_{i,a}, v_{j,b}\} \in E(G)$ if and only if $i \neq j$ and $l_{i,a} = \neg l_{j,b}$. It is easy to see that ϕ is satisfiable if and only if there exists a k-tuple avoiding G.

3. We show that 2-COVER&AVOID can be solved by a polynomial number of calls to a procedure that decides 2-AVOID. Indeed, it is enough to check for each non-edge $\{v_{i,a}, v_{j,b}\}$ whether the graph $G^{\{v_{i,a}, v_{j,b}\}}$ is in 2-AVOID, where $G^{\{v_{i,a}, v_{j,b}\}}$ is obtained from G by replacing the vertices in parts i and j with g_i copies of $v_{i,a}$ and g_j copies of $v_{j,b}$, respectively, as well as copies of their associated edges. A similar argument works for 2-ONE-COVER&AVOID.

4. This is proven by transforming an instance of g-AVOID into an instance of $(g+1)$-ONE-COVER&AVOID which is further transformed into an instance of $(g+2)$-COVER&AVOID; the result follows from the NP-completeness of g-AVOID, for $g \geq 3$, proven in 2. above. Given G, an instance of g-AVOID, build an instance G' for $(g+1)$-ONE-COVER&AVOID in the following way. Append to G one new part indexed by $k+1$ with a vertex $v_{k+1,0}$ plus $g-1$ isolated vertices. Add a new vertex, v_i^A, per part i, $1 \leq i \leq k+1$. Add edges between $v_{k+1,0}$ and each v_i^A, $1 \leq i \leq k$. We need to verify that $G \in g$-AVOID if and only if $G' \in (g+1)$-ONE-COVER&AVOID. By construction, covering $v_{i,a}$ for $1 \leq i \leq k$ while avoiding G' can be easily obtained using a $(k+1)$-tuple T, with $T_i = a$ and $T_\ell = v_\ell^A$ for $\ell \in [1, k+1] \setminus \{i\}$. It is always trivial to similarly cover $v_{k+1,a}$ for $a \neq 0$. Therefore, $G' \in (g+1)$-ONE-COVER&AVOID if and only if we can find a $(k+1)$-tuple T covering $v_{k+1,0}$ and avoiding G', which in turn is equivalent to the first k components of T avoiding G.
Given G, an instance of g-ONE-COVER&AVOID, build G' for $(g+1)$-COVER&AVOID in the following way. Append to G one new part indexed by $k+1$ with a vertex $v_{k+1,0}$ plus $g-1$ isolated vertices. Add a new vertex, v_i^A, per part i, $1 \leq i \leq k+1$. Add edges between $v_{k+1,0}$ and each v_i^A, $1 \leq i \leq k$. We need to verify that $G \in g$-ONE-COVER&AVOID if and only if $G' \in (g+1)$-COVER&AVOID. By construction, covering a pair $\{v_{i,a}, v_{j,b}\}$ for $1 \leq i < j \leq k+1, j \neq 0$ if $b = k+1$, while avoiding G' can be easily obtained using a $(k+1)$-tuple T, with $T_i = a$, $T_j = b$ and $T_\ell = v_\ell^A$ for $\ell \in [1, k+1] \setminus \{i, j\}$. To cover a pair $\{v_{i,a}, v_{k+1,0}\}$ it is necessary to find a $(k+1)$-tuple, T such that $T_i = a, T_j \neq v_j^A$ for any $1 \leq j \leq k$ and $T_{k+1} = 0$ which is equivalent to finding a k-tuple in G that covers the point $v_{i,a}$. Therefore, $G' \in (g+1)$-COVER&AVOID if and only if $G \in g$-ONE-COVER&AVOID.
5. We can reduce from g-COVER&AVOID, by just using N large enough (e.g. the upper bound in Eq. (1)) when deciding whether $\langle G, N\rangle \in g$-CAFEN, and the result follows from 4. □

Bryce and Colbourn have previously shown that 3-AVOID is NP-complete [5]. It has recently been shown that g-CAFEN is NP-complete, for any $g \geq 2$ [26]. The fact that the optimization problem 2-CAFEN is NP-complete contrasts with the feasibility results given in 1 and 3 above.

4. A recursive construction for *CAFE*s

We give a construction that allows us to build a *CAFE* for G, given that we have a *CAFE* for each of $G_1, \ldots, G_s$, a number of disjoint components of G with no edges between them. We also require that if G_i contains a point from a factor, then it contains all points from that factor, thus to each disjoint component G_i we may associate a set of factors F_i which are the factors it covers. A natural application is to take $G_1, \ldots, G_s$ as the non-trivial factor-connected components of G, but it is useful to state the result in its most general form, by not requiring that each G_i be connected or factor-connected. We also need to assume that every point in each factor can be set and still avoid the graph.

Theorem 10. *Let $F_1, \ldots, F_s$ be non-overlapping subsets of factors, that is $F_i \subseteq [1, k]$ and $F_i \cap F_j = \emptyset$ for all $1 \leq i, j \leq s, i \neq j$. Let $G = G_{(g_1, \ldots, g_k)}$ be a graph with subgraphs $G_1, \ldots, G_s$ associated to sets of factors $F_1, \ldots, F_s$, and such that every edge of G lies inside one of the G_i. Assume w.l.o.g. that $[1, l] = [1, k] \setminus (F_1 \cup \cdots \cup F_s)$, that is, $[1, l]$ are factors for which all its corresponding vertices are isolated. Assume also that for each vertex, it is possible to find a test that utilizes that vertex and misses the graph. Let $k_i = |F_i|$, $1 \leq i \leq s$. If the following designs exist:*

1. *P_i: a CAFE$^1(p_i, G_i)$, for each $1 \leq i \leq s$, that is, a covering array of strength 1 avoiding graph G_i with k_i columns and p_i rows;*
2. *A_i: an $a_i \times k_i$ array, such that the array C_i obtained by appending the rows of P_i and A_i is a CAFE$(p_i + a_i, G_i)$, for each $1 \leq i \leq s$;*
3. *P: an MCA$(p; 1, (g_1, \ldots, g_\ell))$ on factors $1, \ldots, \ell$; (possibly empty ($p = 0$) if $\ell = 0$)*
4. *Q: a $q \times \ell$ array, such that the array C obtained by appending the rows of P and Q is an MCA$(p + q; 2, (g_1, \ldots, g_\ell))$;*
5. *M: an MCA$(m; 2, (p_1, \ldots, p_s, p))$, that is a mixed covering with m rows, $s+1$ (or s columns if $p = 0$) columns with alphabets sizes $p_1, \ldots, p_s, p$, respectively.*

Then, there exists a CAFE(n, G) with $n = m + \max\{a_1, \ldots, a_s, q\}$.

Proof. Build an array C with its first set of rows consisting of the A_i's, $1 \leq i \leq s$ and Q, pasted side by side, and adding arbitrary repeated rows to subarrays in order to complete rows up to $\max\{a_1, \ldots, a_s, q\}$. Complete the array C by adding m rows, by substituting each symbol a in a column i of M by row a of P_i or P as appropriate. It is easy to check that C avoids G since the columns of C corresponding to each F_i forms a CAFE for G_i; for this same reason, nonedges within each G_i get covered. Finally, we can check that the remaining nonedges (nonedges outside G_i) get covered in the last m rows and that C has $m + \max\{a_1, \ldots, a_s, q\}$ rows. □

Note that if $G \notin$ COVER&AVOID then some of the ingredients may not exist but in this case neither does a *CAFE*(n, G). Otherwise, since we have assumed that $G \in$ ONE-COVER&AVOID, then $G \in$ COVER&AVOID implies that a *CAFE*(n, G) is necessarily a *CAFE*$^1(n, G)$ and that each component must have a *CAFE*^{1} as well. This insures that all the required ingredients must exist; in the worst case the A_i or Q might be empty if the P_i or P is already a *CAFE* for the relevant subgraphs. Also in the case that the graph, G is empty, this construction becomes a standard product construction for covering arrays with mixed alphabet sizes [32].

In addition to providing a construction of a *CAFE* based on its components, the utility of Theorem 10 can be demonstrated by the following corollary. For instance, if $|E(G)|$ is small with respect to k, the size of a CAFE for G has the same asymptotic growth as the corresponding covering array without graph avoidance, i.e. linear in g and in $\log k$. This is in stark contrast to the fact that larger graphs can force the *CAFE* to grow as $\Theta(k^2)$, as seen in Section 2.

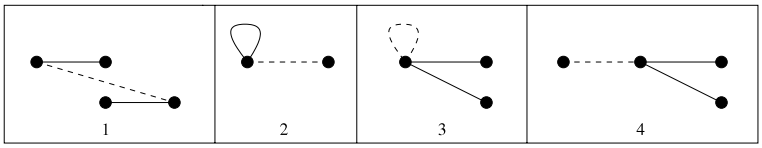


Fig. 2. Forbidden induced subgraphs for binary *CAFE*s.

Corollary 11. *Let $g \geq 2$ be fixed and let $\mathcal{G}$ be a family of consistent graphs such that for all $G \in \mathcal{G}$, $G = G_{k,g}$ and all the edges of G involve factors in F_1 (call $k_1 = |F_1|$ and the graph on factors F_1 is G_1). Then, as $k \to \infty$,*

$$\frac{g}{2}\log_2(k-k_1) \leq CAFEN(G_{k,g}) \leq h(k_1,k) \sim \max\left\{\frac{g^2}{2}k_1^2, \frac{g}{2}\log_2(k-k_1)\right\}. \tag{2}$$

In particular,

1. *$CAFEN(G_{k,g}) = O(k_1^2 + \log k) = O(|E(G_{k,g})|^2 + \log k)$.*
2. *If $k_1 = o(\sqrt{\log k})$ (or $|E(G_{k,g})| = o(\sqrt{\log k})$), then $CAFEN(G_{k,g}) \sim \frac{g}{2}\log_2 k$.*

Moreover, as $k \to \infty$, if $CAFE(G_1) = O(f(k_1))$, then $CAFEN(G_{k,g}) = O(f(k_1) + \log k)$.

Proof. For each $G \in \mathcal{G}$, apply Theorem 10 with $s = 1$. We get that $p_1 \leq gk_1$, since this is the size of the set of points that must be covered by P. Also, since $p = g$, we get $m \leq g^2 k_1$, as $MCAN(2, (p_1, p)) = p_1 p \leq (gk_1)g$. Using the upper bound in Eq. (1), we get $a_1 \leq g^2\binom{k_1}{2} < g^2k_1^2/2$. Finally,

$$q = MCAN(g^{k-k_1}) = CAN(g^{k-k_1}) \sim \frac{g}{2}\log_2(k-k_1),$$

as shown by Gargano, Körner, and Vaccaro [16]. Therefore, as $k \to \infty$, we get

$$\begin{aligned} CAFEN(G) &\leq m + \max\{a_1, q\} \\ &\leq g^2k_1 + \max\left\{g^2k_1^2/2, q\right\} \\ &\sim \max\left\{g^2k_1^2/2, \frac{g}{2}\log_2(k-k_1)\right\}, \end{aligned}$$

which proves the upper bound in Eq. (2). The lower bound follows easily from $CAFEN(G) \geq q \sim \frac{g}{2}\log_2(k-k_1)$. From this we get 2, and also using the fact that G_1 can be chosen so that $k_1 \leq 2|E(G)|$, we get 1. The last statement follows a similar proof as for the upper bound in Eq. (2), but using $O(f(k_1))$ as an upper bound for p_1 and a_1. □

5. An algorithm for constructing binary *CAFE*s

The following proposition characterizes consistent graphs for $g = 2$, that is, graphs with $g = 2$ for which $G = \hat{G}$, or equivalently graphs that are in 2-COVER&AVOID.

Proposition 12. *Let $G = G_{k,2}$ be a graph with vertex set $V(G) = \{v_{i,a} | 1 \leq i \leq k, a \in \{0,1\}\}$. Then, $G = \hat{G}$ if and only if all of the following hold*

1. *$\{v_{i,a}, v_{j,b}\} \in E(G)$ whenever there exist vertices in the same part, $v_{l,c}$ and $v_{l,1-c}$ such that edges $\{v_{i,a}, v_{l,c}\}$ and $\{v_{j,b}, v_{l,1-c}\}$ exist;*
2. *$\{v_{i,a}, v_{j,b}\} \in E(G)$ whenever there is a loop $\{v_{i,a}, v_{i,a}\} \in E(G)$; and*
3. *$\{v_{i,a}, v_{i,a}\} \in E(G)$ (a loop) whenever there exist vertices in the same part, $v_{l,0}$ and $v_{l,1}$ such that edges $\{v_{i,a}, v_{l,0}\}$ and $\{v_{i,a}, v_{l,1}\}$ exist.*

Proof. The three conditions are shown in parts 1–3 of Fig. 2 respectively. In each case the dashed edge is the edge which must be in $E(G)$ when the solid edges are present. The statement is also equivalent to saying that $G = \hat{G}$ if and only if G does not contain the subgraphs in Fig. 2 as induced subgraphs, where only solid edges are present. The necessity is clear, for the dashed lines (non-edges) in Fig. 2 would be non-consistent. The sufficiency is justified next. Let $i \neq j$ and $\{v_{i,a}, v_{j,b}\} \notin E(G)$. We give an algorithm that generates a k-tuple, T, with $T_i = a$ and $T_j = b$, and avoids G. Initially, set $F = \{i, j\}$. At each iteration of the algorithm, T_f has been set for all factors $f \in F$. At each iteration, select $h \in [1,k] \setminus F$. If $v_{h,0}$ forms an edge with v_{f,T_f} for some $f \in F$, then set $T_h = 1$ otherwise set $T_h = 0$. We claim that the set positions in T avoid the graph. This could only fail to avoid G at this iteration if there was $f_1, f_2 \in F$ such that $\{v_{f_1,T_{f_1}}, v_{h,0}\} \in E(G)$ and $\{v_{f_2,T_{f_2}}, v_{h,1}\} \in E(G)$. However, this would imply $\{v_{f_1,T_{f_1}}, v_{f_2,T_{f_2}}\} \in E(G)$ by 1, which is a contradiction to the induction hypothesis that there are no edges involving previously selected positions in T. The algorithm sets $F = F \cup \{h\}$ and continues iterations until $|F| = k$. □

The second and third conditions of Proposition 12 imply another condition which is worth mentioning because it would be used in place of them when focusing on loopless graphs:

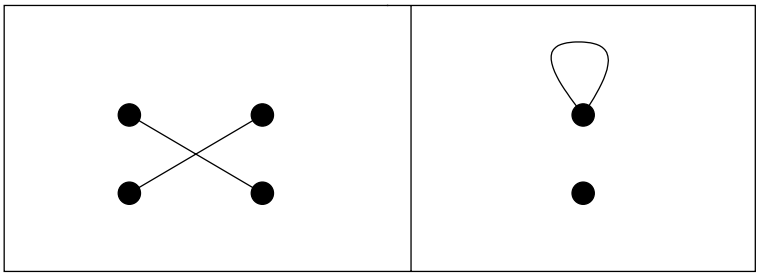


Fig. 3. Configurations for reducing avoidance closed graphs.

4. $\{v_{i,a}, v_{j,b}\} \in E(G)$ whenever there exist vertices in the same part, $v_{l,0}$ and $v_{l,1}$ such that edges $\{v_{i,a}, v_{l,0}\}$ and $\{v_{i,a}, v_{l,1}\}$ exist;

Again this condition is illustrated in part 4 of Fig. 2. The dashed edges in the graphs in Fig. 2 can also be thought of as the edges to add in the *process* of avoidance-closing a graph G.

The avoidance closure of G, $\hat{G}$, can be efficiently computed. Note a directed graph is *transitively closed* if whenever there exists arcs (x, y) and (y, z) then arc (x, z) is present as well.

Proposition 13. *For $G = G(k, 2)$, we can compute $\hat{G}$ in $O(k^2 \times |E(\hat{G})|) = O(k^4)$.*

Proof. Create a directed graph, $\overrightarrow{G}$ from G in the following manner: for every undirected edge $\{v_{i,a}, v_{j,b}\} \in E(G)$, put $(v_{i,a}, v_{j,1-b}), (v_{j,b}, v_{i,1-a}) \in A(\overrightarrow{G})$. It is easy to see that $G = \hat{G}$ (G is avoidance closed) if and only if $\overrightarrow{G}$ is transitively closed, with the convention that if $(v_{i,a}, v_{i,1-a})$ is an arc in $\overrightarrow{G}$ then $(v_{j,b}, v_{i,1-a}), (v_{i,a}, v_{j,1-b}) \in A(\overrightarrow{G})$ for all vertices $v_{j,b} \in V(G)$. From this equivalence, we need to compute the transitive closure of $\overrightarrow{G}$ at most k times, and each one can be done in $O(k|E(\hat{G})|)$ [33]. Translating $\overrightarrow{G}$ back to $\hat{G}$ is done in the following manner: for every contrapositive pair of arcs $(v_{i,a}, v_{j,1-b}), (v_{j,b}, v_{i,1-a})$ in $(\overrightarrow{G})$ put undirected edge $\{v_{i,a}, v_{j,b}\} \in E(\hat{G})$. This can be done in linear time in $|E(\hat{G})|$. □

Once a graph, G, has been avoidance-closed, if it contains any of the subgraphs in Fig. 3, then one of its factor can be removed without affecting the existence or the size of a $CAFE(n, G)$. In the first case, the choice of vertices in the second factor is equivalently dependent on the choice of vertex from the first factor, so the non-edges between these two factors can be contracted, decreasing the number of factors by one. In second case, the choice of vertex from the factor is forced, and this factor can be removed. The corresponding column in a $CAFE(n, G)$ can be fully reconstructed later. The details of these reduction procedures are discussed in the proof of Theorem 17.

Definition 14. Let $G = G_{k,2}$ be such that $G = \hat{G}$. G is said to be *reduced* if it has no subgraphs of the form shown in Fig. 3. If G is not reduced, it can be *reduced* by removing factors whose choices are forced or redundant.

For the rest of this section we assume that graphs are reduced but we explicitly include the reduction procedure in Algorithm 1.

Our next result is that for $g = 2$, a *CAFE* for a graph corresponds to an edge cover by cliques of its complement, i.e. for $g = 2$ the k-uniform restriction on the edge clique cover does not need to be explicitly imposed.

Theorem 15. *Let $G = G_{k,2}$ be such that $G = \hat{G}$. Then, $CAFEN(G) = \theta'(\overline{G^{|}})$.*

Proof. Since every row of a $CAFE(n, G)$ must be an independent set of size k in $G^{|}$, the maximum size of an independent set is k and every non-edge of $G^{|}$ must be covered by some row of the $CAFE(n, G)$, it is easy to see that $\theta'(\overline{G^{|}}) \leq CAFEN(G)$. Given any clique in $\overline{G^{|}}$ (independent set in $G^{|}$), we show that it can be extended to a clique of size k in $\overline{G^{|}}$. Suppose that the vertices $\{v_{i_1,a_1}, v_{i_2,a_2}, \ldots, v_{i_{k'},a_{k'}}\}$, $k' < k$, induce a clique in $\overline{G^{|}}$. Suppose that neither vertex for part j can be added and still induce a clique. This means that some edge $\{v_{i_l,a_l}, v_{j,0}\}$ and $\{v_{i_m,a_m}, v_{j,1}\}$ must both be missing from $\overline{G^{|}}$, or in other words that these two edges are present in $G^{|}$. If $l = m$ and $a_m = a_l$ then vertex v_{i_l,a_l} would have been removed to produce a reduced G. Similarly if $l = m$ and $a_m \neq a_l$ then one of factor j or l is redundant and would have been removed to produce a reduced G. If $l \neq m$ then since $G = \hat{G}$, edge $\{v_{i_l,a_l}, v_{i_m,a_m}\}$ must appear in G, this contradicts the fact that this was an edge in a clique in $\overline{G^{|}}$. Hence, every clique can be extended by one point and induction shows that all maximal cliques are of size k. This establishes that $CAFEN(G) \leq \theta'(\overline{G^{|}})$. □

Using the lower bound on $\theta'(G)$ from [18] we get:

Corollary 16. *Let $G = G_{k,2}$ be such that $G = \hat{G}$ and let $k' \leq k$ be the number of factors after G is reduced. Then*

$$CAFEN(G) \geq \log_2(2k' + 1).$$

Algorithm 1 Build $CAFE(n, G)$ for $g = 2$.

Require: $G = G_{k,2}$ and G avoidance closed.
  procedure BUILDCAFE(G) **** (main procedure) ****
    $CAFE \leftarrow \emptyset$
    if $G \neq K_{k,2}$ **then**
      Using $2 - SAT$ formulation find an independent set of size k, $I \in G^{|}$
      Swapping values in each factor if necessary, let $I = \{v_{i,0} | 1 \leq i \leq k\}$
      Remove factors that contain a loop
      Contract pairs of factors with parallel edges
      Order the factors so all 0–1 edges go down and to the right ▷ Topologically sort $\overrightarrow{G}$
      $V(G_B) \leftarrow V(G)$; $E(G_B) \leftarrow \{\{v_{i,0}, v_{j,1}\} \in E(G) : i, j \in [1, k]\}$
      $V(G_1) \leftarrow \{v_{i,1} \in V(G) : i \in [1, k]\}$; $E(G_1) \leftarrow \{\{v_{i,1}, v_{j,1}\} \in E(G) : i, j \in [1, k]\}$
      $B \leftarrow$ BUILDBIPARTITE(G_B)
      $C \leftarrow$ BUILDONES(G_1)
      $CAFE \leftarrow \{0^k\} \cup B \cup C$
      for all i, a factor contracted to j **do**
        add new column i in $CAFE$
        set values in column i as forced by values in column j
      for all i, a factor with a loop **do**
        add new column i in $CAFE$
        set values in column i as forced by loop.
      Relabel values that have been swapped in each factor.
    return $CAFE$
  procedure BUILDBIPARTITE(G)
    for all $1 \leq i \leq k$ **do**
      for all $k \geq j > i$ **do**
        $b_j \leftarrow 0$
      $b_i \leftarrow 1$
      for all $i > j \geq 1$ **do**
        if $\{v_{j,0}, v_{i,1}\} \in E(G)$ **then**
          $b_j \leftarrow 1$
        else
          $b_j \leftarrow 0$
      $B \leftarrow B \cup \{b\}$
    return B
  procedure BUILDONES(G)
    edge-cover $\overline{G}$ by cliques $(K^1, K^2, \ldots, K^m)$
    for all $1 \leq i \leq m$ **do**
      for all $j \in V(K^i)$ **do**
        $c_j \leftarrow 1$
      for all $j \notin V(K^i)$ **do**
        if $\{v_{j,0}, v_{l,1}\} \in E(G)$ for some $l \in V(K^i)$ **then**
          $c_j \leftarrow 1$
        else
          $c_j \leftarrow 0$
      $C \leftarrow C \cup \{c\}$
    return C

Since the extension of cliques to k-cliques can be done greedily (see proof above), any heuristics or approximation algorithms for the edge clique covering problem adapt directly to produce $CAFE(n, G)$ for $g = 2$. Additionally we now know that for arbitrary graph G, there exists a $G' = \hat{G}' = G'_{k,2}$ such that $\theta'(G) + 2 = CAFEN(G')$ [26]. This result allows us to translate any hardness of approximation results known for $\theta'(G)$ into one for $CAFEN(G_{k,2})$.

In Algorithm 1, we give a method that builds a $CAFE$ for a graph $G = G_{k,2} = \hat{G}$. This algorithm reduces the problem to finding an edge covering by cliques of a subgraph of $\overline{G^{|}}$. At a first glance, this might not seem much better than what Theorem 15 gives us. However, in certain situations the subgraph obtained may be much simpler, as in the case of bipartite G (see Corollary 20).

Theorem 17. *Let $G = G_{k,2}$ be such that $G = \hat{G}$ and let $k' \leq k$ be the number of factors after G is reduced. Let G_1 be the graph calculated in* BUILDCAFE*, which is obtained by removing all edges incident to the vertices corresponding to a k-tuple avoiding G.*

Then, Algorithm 1 is correct, and BuildCafe *returns a CAFE*(n, G) *for* $n = k' + 1 + m \leq k + 1 + m$, *where* m *is the size of an edge clique cover of* $\overline{G_1}$.

Proof. If the reduction of G is $K_{k',2}$ then this is equivalent to $G = K_{k,2}$ because of closure. So if $G = \hat{G} = K_{k,2}$ then the required *CAFE* is empty. So from now on we assume that $G = \hat{G} \neq K_{k,2}$. Since $G = \hat{G} \neq K_{k,2}$, there exists a feasible row via the reduction from 2-AVOID to 2-SAT which yields a k-tuple T (test) that avoids G. Thus, we relabel the values of 0 and 1 in each factor so that $T = (0, \ldots, 0)$. After relabeling, all edges of G are between a vertex valued 0 (i.e. $v_{i,0}$ for some i) and a vertex valued 1 or are between two vertices valued 1.

Next, we reduce the graph. We remove any factors which have their values forced. This happens to factor i if and only if there is a loop at $v_{i,1}$. Note that after we are done building a *CAFE* for the graph without these factors, we simply put these factors back into the array as new columns and set their value to 0 in every row. Next, we iteratively contract any pair of factors which contain two *parallel edges* between them, that is two edges of the form $\{v_{i,0}, v_{j,1}\}$ and $\{v_{i,1}, v_{j,0}\}$. These edges imply that the value in factor i must be equal to the value in factor j. Note that after we are done building a *CAFE* for the graph with contracted factors, we simply put the extra factors back into the array as new columns and set to their forced value in every row.

We can now assume that our graph is avoidance closed, has no loops and no pair of parallel edges and thus at most one edge between any two factors. It has no 0–0 edges and thus only 0–1 edges and 1–1 edges. We reorder the factors so that all 0–1 edges slope down and to the right, that is, if $\{v_{i,0}, v_{j,1}\} \in E(G)$ then $j > i$. This is always possible and can be verified using the fact that there is at most one edge between any two factors and that $G = \hat{G}$ (alternatively, it is equivalent to the fact that the corresponding directed graph becomes acyclic and can be topologically sorted).

We edge-decompose the graph into the subgraphs containing exclusively each of these two kinds of edges: the *bipartite subgraph* (G_B) and the *1–1 subgraph* (G_1). We now construct three partial covering arrays each of which avoids the whole graph. They will cover the non-forbidden 0–0 pairs, the 0–1 pairs and the 1–1 pairs, respectively. The first covering array, A, is a single row containing only 0's.

The second partial covering array, B, will have exactly k' rows with the values given in the array below already set and for the remaining range of entries, $i > j$, $B_{i,j} = 1$ if $\{v_{j,0}, v_{i,1}\} \in E(G_B)$ and 0 otherwise:

$$\begin{array}{cccccccc}
1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\
 & & 1 & 0 & \cdots & 0 & 0 & 0 \\
 & & & 1 & & 0 & 0 & 0 \\
 & & & & \ddots & & \vdots & \\
 & & & & & 1 & 0 & 0 \\
 & & & & & & 1 & 0 \\
 & & & & & & & 1
\end{array}$$

We must show that this covers all required 0–1 pairs and avoids all forbidden 0–1 and 1–1 edges. For every pair of factors $i < j$, we must always cover the pair $\{v_{i,1}, v_{j,0}\}$ and this is covered in row i. If we also need to cover the pair $\{v_{i,0}, v_{j,1}\}$ this is done in row j; $B_{j,i} = 0$ since this pair is not a forbidden edge.

We now check that we do not cover any forbidden edges. Suppose $B_{\ell,i} = 0$ and $B_{\ell,j} = 1$ for $i < j < l$ (by the construction of B, this range is the only place such an forbidden edge could be covered). By construction the pair $\{v_{i,0}, v_{\ell,1}\}$ is not an edge of G, $\{v_{j,0}, v_{\ell,1}\} \in E(G)$ and by closure, $\{v_{i,0}, v_{j,1}\}$ is not an edge of G. Similarly, suppose $B_{\ell,i} = 1$ and $B_{\ell,j} = 1$ for $i < j < l$. By construction, both $\{v_{i,0}, v_{\ell,1}\}$, $\{v_{j,0}, v_{\ell,1}\} \in E(G)$. If $\{v_{i,1}, v_{j,1}\}$ was a forbidden edge then by the existence of these three edges and closure we get that $\{v_{j,1}, v_{l,1}\} \in E(G)$ and thus factor l was removed because its value was forced to be 0. Thus $\{v_{i,1}, v_{j,1}\}$ is not a forbidden edge.

The third partial covering array, C, will have one row for each clique in an edge clique cover of the complement of G_1 (only on the vertices at the 1-level). These cliques in the complement are independent sets in G_1 and thus avoid the 1–1 edges and every 1–1 pair that is not an edge will be covered. These are also cliques in the graph $\overline{G^{|}}$ and thus by the proof of Theorem 15 they can be extended to maximal cliques of size k', which is equivalent to completing this row of C while avoiding G. □

Next, we want to apply the algorithm to bipartite graphs. We need the following lemma.

Lemma 18. *Let* $G = \hat{G}$. *Then* G *is bipartite if and only if* $G^{|}$ *is bipartite.*

Proof. We first note that the "if" part of this lemma is trivial so we need only prove the "only if". Since G is bipartite it has a 2-coloring $\{T, F\}$ of its vertices. We claim that there is a colouring which which has the desired property of colouring the two points in a factor differently. We note that since G is bipartite, it has no odd cycles.

Suppose that there exists a factor, j, where both vertices are forced to have the same color. This would mean that there is an even length path in G from $v_{j,0}$ to $v_{j,1}$. Since there exists at least one even length path connecting the two vertices of some factor, let $v_{i,0} = q_0 q_1 q_2 \ldots q_{2r+1} q_{2r+2} = v_{i,1}$ be the shortest even length path between two vertices of the same factor. Thus $q_0 q_1$ and $q_{2r+2} q_{2r+1}$ are edges of G, so by closure $q_1 q_{2r+1}$ is an edge of G. However, this implies $q_1 q_2 \ldots q_{2r+1}$ is an odd cycle

in G, which contradicts the assumption that G is bipartite. So G has a 2-coloring in which $v_{i,0}$ and $v_{i,1}$ get different colors for every factor i. □

From this proof we can see that the two colour classes are equivalent to two disjoint rows of a *CAFE*, thus we get the next result.

Corollary 19. *G is bipartite if and only if it admits a CAFE(n, G) which contains two disjoint rows.*

Corollary 20. *Let $G = G_{k,2}$, $G = \hat{G}$ and suppose that G is bipartite. Let $k' \leq k$ be the number of factors after G is reduced, then $CAFEN(G) \leq k' + 2 \leq k + 2$.*

Proof. Since G is bipartite, so is $G^{|}$ by Lemma 18. Thus, we can choose the k'-tuple avoiding G in Theorem 17, as one of the two parts of $G^{|}$, which yields G_1 with no edges and $m = 1$. Thus the *CAFE* produced by Algorithm 1 has size $k' + 2$. □

We know that the graph G which has edges $\{v_{i,0}, v_{j,1}\}$ for all $1 \leq i < j \leq k$ has $CAFEN(G) = k + 1$ so the upper bound given by this algorithm is close to best possible in this case.

If every factor connected component of G is bipartite then using Theorem 10 we get:

Corollary 21. *Let $G = G_{k,2}$, $G = \hat{G}$ and let $k' \leq k$ be the number of factors after G is reduced. Suppose the factor connected components of the reduction of G are $\{G_1, G_2 \ldots G_s\}$ with k'_i factors in G_i. Assume also that each G_i is bipartite. Then*

$$CAFEN(G) \leq CAN(s, 2) + \max_{1 \leq i \leq s}\{k'_i\} - 1 \in O(\log s + \max_{1 \leq i \leq s}\{k'_i\}).$$

Proof. Algorithm 1 produces P_i which all have size 2 and can be used in Theorem 10. □

For the case of non-bipartite graphs, Algorithm 1 requires us to build an edge clique cover of $\overline{G_1}$. In this case, we can use one of the constructive results on edge clique covers listed in Section 2. We hope that an edge clique cover of $\overline{G_1}$ will be much smaller than one of $\overline{\hat{G}^{|}}$, as it was in the case of bipartite G. A direction of further research is to extend the results obtained for bipartite graphs to other interesting classes of graphs.

References

[1] Robert C. Brigham, Ronald D. Dutton, Upper bounds on the edge clique cover number of a graph, Discrete Math. 52 (1) (1984) 31–37.
[2] Robert C. Brigham, Ronald D. Dutton, A compilation of relations between graph invariants, Networks 15 (1) (1985) 73–107.
[3] Robert C. Brigham, Ronald D. Dutton, Changing and unchanging invariants: The edge clique cover number, in: Proceedings of the Twentieth Southeastern Conference on Combinatorics, Graph Theory, and Computing, Boca Raton, FL, 1989, vol. 70, 1990, pp. 145–152.
[4] Robert C. Brigham, Ronald D. Dutton, A compilation of relations between graph invariants. Supplement I, Networks 21 (4) (1991) 421–455.
[5] R.C. Bryce, C.J. Colbourn, Prioritized interaction testing for pairwise coverage with seeding and constraints, Inf. Softw. Technol. 48 (2006) 960–970.
[6] K. Burr, W. Young, Combinatorial test techniques: Table-based automation, test generation, and code coverage, in: Proc. Intl. Conf. on Soft. Test. Anal. and Rev., ACM, New York, 1998, pp. 503–513.
[7] J.N. Cawse, Experimental design for combinatorial and high throughput materials development, GE Glob. Res. Tech. Rep. 29, 2002, pp. 769–781.
[8] M.B. Cohen, M.B. Dwyer, J. Shi, Interaction testing of highly-configurable systems in the presence of constraints, in: International Symposium on Software Testing and Analysis, ISSTA, London, 2007, pp. 129–139.
[9] Charles J. Colbourn, Combinatorial aspects of covering arrays, Matematiche (Catania) 59 (1–2) (2006) 125–172.
[10] Charles J. Colbourn, Covering arrays, in Colbourn and Dinitz [11], pp. 361–364.
[11] Charles J. Colbourn, Jeffrey H. Dinitz (Eds.), Handbook of Combinatorial Designs, second ed., in: Discrete Math. and its Applications (Boca Raton), Chapman & Hall/CRC, Boca Raton, FL, 2007.
[12] S.R. Dalal, A.J.N. Karunanithi, J.M.L. Leaton, G.C.P. Patton, B.M. Horowitz, Model-based testing in practice, in: Proc. Intl. Conf. on Software Engineering, ICSE 99, 1999, pp. 285–294.
[13] P. Danziger, E. Mendelsohn, L. Moura, B. Stevens, Covering arrays avoiding forbidden edges, in: Proc. COCOA 2008 Second International Conference on Combinatorial Optimization and Applications, Lecture Notes in Computer Science, 5165, St. Johns, NL, 2008, pp. 296–308.
[14] S. Dunietz, W.K. Ehrlich, B.D. Szablak, C.L. Mallows, A. Iannino, Applying design of experiments to software testing, in: Proc. Intl. Conf. on Software Engineering, ICSE 97, IEEE, Los Alamitos, CA, 1997, pp. 205–215.
[15] Paul Erdős, A.W. Goodman, Lajos Pósa, The representation of a graph by set intersections, Canad. J. Math. 18 (1966) 106–112.
[16] L. Gargano, J. Körner, U. Vaccaro, Sperner capacities, Graphs Combin. 9 (1) (1993) 31–46.
[17] Oliver Goldschmidt, Dorit S. Hochbaum, Cor Hurkens, Gang Yu, Approximation algorithms for the k-clique covering problem, SIAM J. Discrete Math. 9 (3) (1996) 492–509.
[18] A. Gyárfás, A simple lower bound on edge coverings by cliques, Discrete Math. 85 (1) (1990) 103–104.
[19] Alan Hartman, Leonid Raskin, Problems and algorithms for covering arrays, Discrete Math. 284 (1–3) (2004) 149–156.
[20] Minmei Hou, Piotr Berman, Louxin Zhang, Webb Miller, Controlling size when aligning multiple genomic sequences with duplications, in: Algorithms in Bioinformatics, in: Lecture Notes in Comput. Sci., vol. 4175, Springer, Berlin, 2006, pp. 138–149.
[21] L.T. Kou, L.J. Stockmeyer, C.K. Wong, Covering edges by cliques with regard to keyword conflicts and intersection graphs, Comm. ACM 21 (2) (1978) 135–139.
[22] D.R. Kuhn, M. Reilly, An investigation of the applicability of design of experiments to software testing, in: Proc. 27th Annual NASA Goddard/IEEE Software Engineering Workshop, IEEE, Los Alamitos, CA, 2002, pp. 91–95.
[23] D.R. Kuhn, D.R. Wallace, A.M. Gallo, Software fault interactions and implications for software testing, IEEE Trans. Soft. Eng. 30 (2004) 418–421.
[24] L. Lovász, On covering of graphs, in: Theory of Graphs (Proc. Colloq., Tihany, 1966), Academic Press, New York, 1968, pp. 231–236.
[25] Carsten Lund, Mihalis Yannakakis, On the hardness of approximating minimization problems, J. Assoc. Comput. Mach. 41 (5) (1994) 960–981.
[26] Elizabeth Maltais, Lucia Moura, Finding the best CAFE is NP-hard, extended abstract, September 2009 (submitted).
[27] R. Mandl, Orthogonal latin squares: An application of experiment design to compiler testing, Comm. ACM 28 (1985) 1054–1058.
[28] C. Martinez, L. Moura, D. Panario, B. Stevens, Algorithms to locate errors using covering arrays, in: Proc. LATIN'2008 — 8th Latin American Theoretical Informatics Conference, in: Lecture Notes in Comput. Sci., vol. 4957, April 2008, pp. 504–519.
[29] C. Martinez, L. Moura, D. Panario, B. Stevens, Locating errors using ELAs, covering arrays and adaptive testing algorithms, SIAM J. Discrete Math., (in press).
[30] Karen Meagher, Lucia Moura, Latifa Zekaoui, Mixed covering arrays on graphs, J. Combin. Des. 15 (5) (2007) 393–404.

[31] Karen Meagher, Brett Stevens, Covering arrays on graphs, J. Combin. Theory Ser. B 95 (1) (2005) 134–151.
[32] Lucia Moura, John Stardom, Brett Stevens, Alan Williams, Covering arrays with mixed alphabet sizes, J. Combin. Des. 11 (6) (2003) 413–432.
[33] E. Nuutila, Efficient transitive closure computation in large digraphs, in: Acta Polytechnica Scandinavica, in: Mathematics and Computing in Engineering Series, vol. 74, Finnish Academy of Technology, 1995.
[34] James Orlin, Contentment in graph theory: Covering graphs with cliques, Indag. Math. 39 (5) (1977) 406–424.
[35] A.H. Ronneseth, C.J. Colbourn, Merging covering arrays and compressing multiple sequence alignments, Discrete Appl. Math. 157 (2009) 2177–2190.
[36] Gadiel Seroussi, Nader H. Bshouty, Vector sets for exhaustive testing of logic circuits, IEEE Trans. Inform. Theory 34 (3) (1988) 513–522.
[37] D.E. Shasha, A.Y. Kouranov, L.V. Lejay, M.F. Chou, G.M. Coruzzi, Using combinatorial design to study regulation by multiple input signals: A tool for parsimony in the post-genomics era, Plant Physiol. 127 (2001) 1590–2594.